Presented at Intelligent Environments 2011 (IE'11), Nottingham 27-29th July 2011 (pre-review version)

Innovative Locking in AmI

Efficiently Removing Instabilities in Multi Agent Systems

Emmanuel Amaro
Department of Engineering and Science
Monterrey Tech Campus Leon
Leon Guanajuato Mexico
emmanuelamaro@gmail.com

Victor Zamudio, Rosario Baltazar and
Miguel Angel Casillas
Department of Postgraduate Studies and Research
Leon Institute of Technology
Leon Guanajuato Mexico
vic.zamudio@ieee.org

Vic Callaghan
School of Computer Science and Electronic Engineering
University of Essex
Wivenhoe Park, United Kingdom
vic@essex.ac.uk

***Abstract*—Cyclic instabilities can impact the performance of a multi agent system, especially in terms of the user's point of view. Different strategies can be use in order to prevent this problem. In this paper we present two strategies, ONL1 and ONL2 that aim at minimizing the collateral consequences of locking. These two strategies focus on minimizing the number of nodes locked, and also the total weight. These strategies performed better than the current strategy, INPRES, especially in very dense systems.**

Keywords-component; cyclic instability; locking; multiagent ststems; complexity.

1. Introduction

Ambient Intelligence and in particular rule-based multi agent systems have been found to suffer from a fundamental problem of cyclic instability, rooted in rule based interaction between agents. Circular dependencies arising from agent rules are a necessary condition for this behaviour; however, other aspects should be taken into account, such as the rules themslves, and the initial conditions of the system. One solution that has been reported to solve this condition is called INPRES (Instability Prevention System) and is based on locking agent actions [15, 16, 17]. However, this strategy can impact noticeably on the services provided to the user, as the flux of information throughout the system is diminished. In this paper we propose an innovative refinement to INPRES called Optimized Node Locking ONL that aims to minimize the number of agents locked choosing those with less importance on the network.

2. Theoretical background

2.1 Interaction Networks and Agents

Interaction Network (IN) is a digraph (V, E) in which the vertex $v_k \in V$ is a pervasive intelligent device or agent A_k and $(v_i, v_j) \in E$ if the Boolean functions φ_j or ψ_j of the pervasive intelligent device A_j depends on the state s_i of the device A_i. An example of an Interaction Network can be seen of Fig. 1. Interaction Networks are able to represent the topological properties of the system. In particular, the presence of feedback or loops in the system is a necessary condition for the instabilities to emerge.

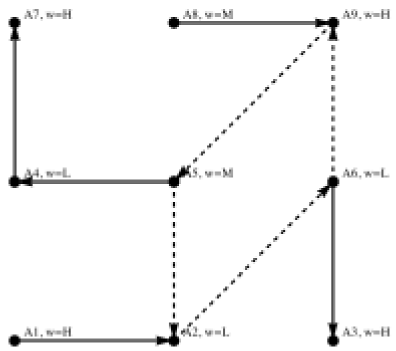

Figure 1 - An Interaction Network showing a loop in dashed lines.

An agent A_k is an autonomous device consisting of a triplet $[s_k, r_k, w_k]$ where k is the agent number for $k = 1,2,3,..n$, with n being the total agents number and:

s_k: is the binary state of the *k*-agent defined over $\{0,1\}$
w_k: is the importance or weight over $\{Low, Medium, High\}$
r_k: is the set of Boolean rules of the *k*-agent $\{\varphi_k, \psi_k\}$ defined as:

$$\text{If } \varphi_k \text{ then } s_k = 1 \tag{1}$$

The authors want to thank ITESM Campus León and ITL for their support to this research.

$$\text{If } \psi_k \text{ then } s_k = 0 \tag{2}$$

With

$$\varphi_k, \psi_k : S \to \{0,1\} \tag{3}$$

If we have n autonomous devices $A_1, A_2, \ldots, A_n$ the state of the system is $S = (s_1, s_2, \ldots, s_n)$.

The rules defined in (1) and (2) are consistent in the sense that $\varphi_k = \psi_k^{-1}$. With this, the case of contradictory rules (e.g. one device ending up with two different states simultaneously) is avoided.

The set of rules defined over the agents can be used to build a network capturing the functional dependencies between the agents, as will be shown in the next section.

The factor of importance corresponds to the inherent weight of the agent, taking into account the following aspects [18]: inherent importance (devices can have different importance according to the services or functionality provided) and user's preferences (users could have different preferences). As it can be seen, this model is very similar to a state machine, in particular, Boolean networks [18]. However, in the case of Boolean networks the rules are homogeneous, and the connections are symmetric and time-independent.

Based on these topological properties on the digraph, different strategies can emerge. In particular, the strategy based on locking a set of agents with less connectivity has been proven to be effective [15, 16]. However, in the case of complex topologies and in particular with coupled loops i.e., with common vertex between loops, this strategy (Instability Prevention System-INPRES) tends to overlock the system, as for each loop or feedback circuit found in the IN, there is a locked agent. Another strategy c-INPRES [18] is based on analysing local rules of coupled agents (ie belonging to two or more cycles). The strategy presented on this paper does not analyze rules, and therefore is more general and easier to implement.

3. Optimized Node Locking

In this paper we introduce two new algorithms, ONL1-INPRES or ONL1, and ONL2-INPRES or ONL2 for future reference. These algorithms are a further refinement of INPRES which aims to solve the problem of cyclic instability. The main advantage of these algorithms is that they approach the problem in a more general way, locking nodes to achieve stability in the system while minimizing the number of locked nodes and the total sum of weights of locked nodes. In the same way, ONL1 and ONL2 don't search for, or expect, certain topologies, properties or the formation of specific rules in the system. Thus, they will perform efficiently in any kind of environment, however, it should be stated that they work best in very dense and coupled environments, where their benefits are amplified.

3.1 ONL1-INPRES

ONL1-INPRES is the acronym for Optimized Node Locking for the Instability Prevention System. As a general overview, the algorithm will remove the instability from the system by locking a node for each cycle. Just after a node is locked, the algorithm will search for the same node in the remaining cycles and if it is found, that cycle is marked also as stable. This way, the algorithm tries to maximize the effect of locking a node.

Because the environment in which the algorithm is designed to perform is expected to be very dense and coupled, the number of locked nodes needed to achieve stability decreases dramatically in comparison to INPRES.

```
   onl1(Graph g)
1      cycles = findCycles(g)
2      ∀ cycle Є cycles
3         stableCycles.add(cycle)
4         cycle.findMinWeightNode()
5         cycle.lockedNode(cycle.minNode)
6         ∀ insCycle Є cycles
7            if (cycle.minNode Є insCycle)
8               insCycle.lockedNode(cycle.minNode)
9               stableCycles.add(insCycle)
10              cycles.remove(insCycle)
```

Considering the above, line 1, the function findCycles(g) is a modified version of the Depth First Search (DFS) algorithm. We want to iterate over all the cycles that the DFS found (line 3), except for those that have been found already as stable (line 9). In line 3 we add the current cycle to a set of stable cycles. In line 4 and 5 we find the node with the minimum weight of the cycle, and then we lock that node (hence, the cycle can no longer perturb the system). After locking the minimum node, we search the graph for that node. If it is found, the cycle that has it is also in a stable state, so we mark the node of the cycle as locked (line 8), add the cycle to the set of stable cycles (line 9) and remove that cycle from the ones the algorithm needs to go through (line 10). In this way, we considerably need less locked nodes to bring the system to a stable state.

3.2 ONL2-INPRES

This algorithm is similar to ONL1-INPRES. The main difference is that this one will try to lock the least weighted nodes first with the objective of diminishing the overall sum of weights in locked nodes.

```
onl2(Graph g)
    cycles = findCycles(g)
    ∀ cycle Є cycles
       cycle.findMinWeightNode()
    sort(cycles)
    ∀ cycle Є cycles
       stableCycles.add(cycle)
       cycle.lockedNode(cycle.minNode)
       ∀ insCycle Є cycles
          if (cycle.minNode Є insCycle)
             insCycle.lockedNode(cycle.minNode)
             stableCycles.add(insCycle)
             cycles.remove(insCycle)
```

In line 2 and 3 we can see that we are first finding the least-weighted nodes, so that we can sort the cycles later on (line 4). The Quicksort algorithm was used to sort the cycles, using the minimum weighted node of each as the comparison value; the sorting is ascendant. The rest of the algorithm is almost the same as ONL1; with the exception that line 5 of first algorithm is not present in this one.

Even though the algorithms are similar, they lead us to some conclusions that we did not expect, so we chose to include both in this paper. Those conclusions will be explained later in the discussion.

4. Experimental results

4.1 Experiment 1 - coupled in one point

Several experiments were performed, using well known benchmarks reported in [17]. The first topology tested had 64 nodes, each with coupled cycles in one point, as shown in Fig. 3.

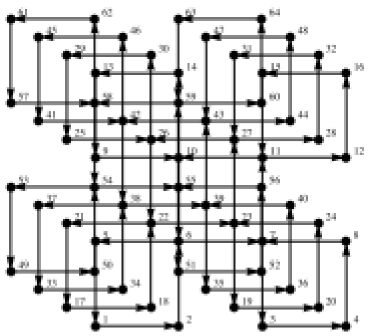

Figure 2 – Topology of experiment 1

The following graph shows that the system is currently in an instable state.

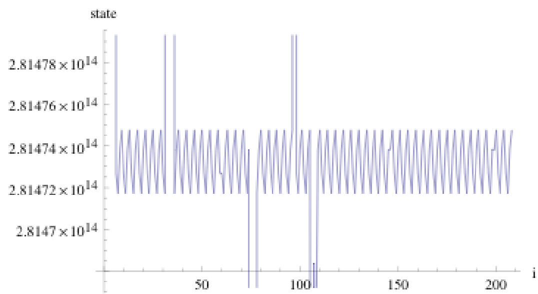

Figure 3 – Initial stability experiment 1

After running the algorithms INPRES, ONL1-INPRES and ONL2-INPRES with the current experimental configuration, the locking vector produced by each algorithm are:

INPRES	{0,1,0,1,1,0,1,1,0,1,1,0,1,1,1,1,0,1,0,1,1,0,1,1,0,1,1,0,1,1,1,1,0,1, 0,1,1,0,1,1,0,1,1,0,1,1,1,1,0,1,0,1,1,0,1,1,0,1,1,0,1,1,1,1}
ONL1-INPRES	{1,0,0,0,0,0,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,0,0,0,0,0,1,0, 0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0}
ONL2-INPRES	{0,0,0,0,0,1,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,0,0,0,0,0,1,0, 0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0}

Table 1 – Resulted locking vectors per algorithm, experiment 1

To following graphs show the algorithms are achieving stabilization of the system:

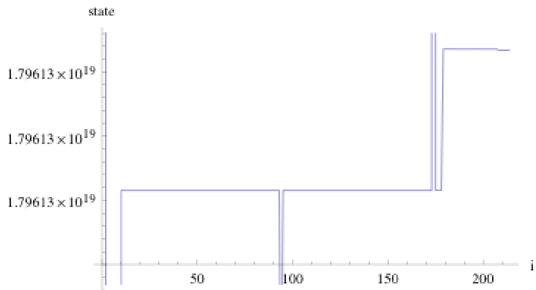

Figure 4 - Stability with ONL1-INPRES

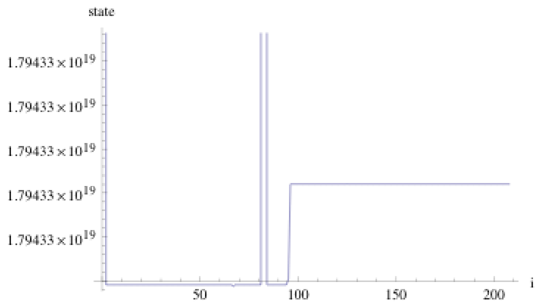

Figure 5 – Stability with ONL2-INPRES

In this experiment, INPRES locked 44 nodes whereas ONL1-INPRES and ONL2-INPRES both locked 16 nodes (less than half). The sum of the weights of the locked nodes in INPRES was 253 and both new algorithms summed 44 in total. This shows an advantage for the new algorithms in the number of locked nodes and clearly allows a stable and less-disabled system.

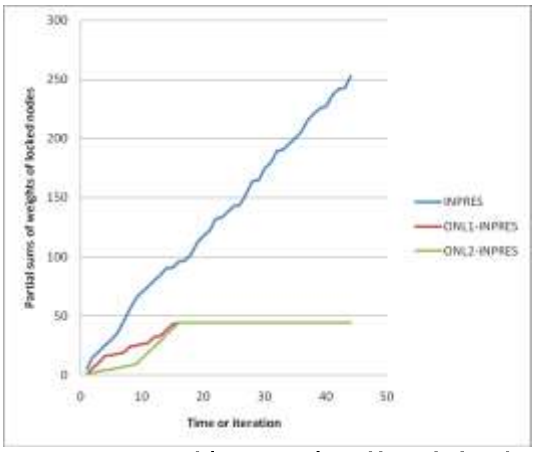

Figure 6 – System modifications performed by each algorithm

The previous graph shows how each algorithm modifies the system in which these are run. Both ONL1 and ONL2 stopped modifying the system after locking the 16th node, while

INPRES kept modifying the environment until it locked its 44th node. This graph shows the first noticeable difference between ONL1 and ONL2: We can observe that ONL2 begins by choosing the bests nodes to lock based on their weights and resulting in an initial lower partial sum. Nevertheless, when both algorithms finish locking their respectives nodes, the sum of weights of locked nodes are equal.

The remaining experiments will show that ONL2 will not be able to produce a total lower sum of weights of the locked nodes in comparison to ONL1. ONL2 will get the same total sum that ONL1, and in some cases, a little bit higher. This is not a behaviour we expected since ONL2 was designed from the beginning to produce lower total sums of weights of locked nodes. This result and its complexity will be addressed later in this paper.

4.2 Experiment 2 - coupled in two points

This experiment was used a system with nodes coupled in two points, as shown in fig. 8.

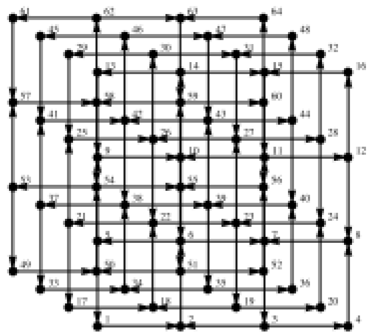

Figure 7 – Topology of experiment 2

Again, in the following graph we can see that the system is not stable.

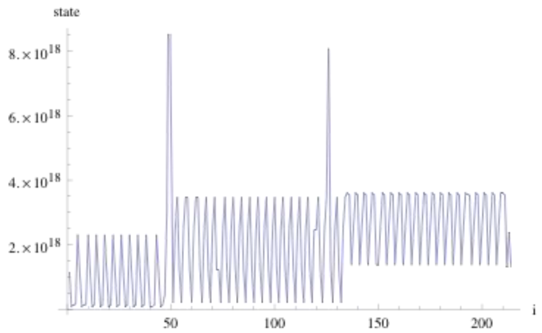

Figure 8 – Original unstable system of experiment 2

The following table shows the locking vector of each algorithm.

INPRES	{0,0,1,0,1,0,1,0,0,1,1,1,1,1,0,1,0,0,1,0,1,0,1,0,0,1,1,1,1,1,0,1,0,0, 1,0,1,0,1,0,0,1,1,1,1,1,0,1,0,0,1,0,1,0,1,0,0,1,1,1,1,1,0,1}
ONL1-INPRES	{1,0,0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,1,0,0,1,0,0,0,1,1,0,0,0,0,0,1,0, 1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,1,0,1,0,0,1,0,0,0,0,1,0}
ONL2-INPRES	{1,0,0,0,0,1,1,0,1,0,0,1,0,1,0,0,0,0,1,0,0,1,0,0,0,1,1,0,0,0,0,0,1,0, 1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,1,0,1,0,0,1,0,0,0,0,1,0}

Table 2 - Resulted locking vectors per algorithm, experiment 2

The following graphs show the stabilization achieved by ONL1 and ONL2.

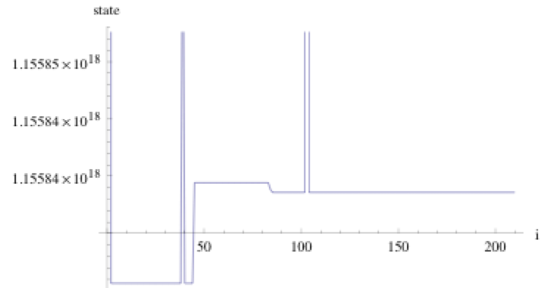

Figure 9 - Stability with ONL1-INPRES

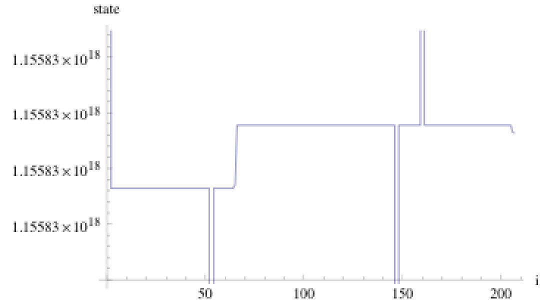

Figure 10 - Stability with ONL2-INPRES

INPRES locked 36 nodes, whereas ONL1-INPRES and ONL2-INPRES locked 18 and 19 nodes respectively. The sum of the weights of the locked nodes in INPRES was 176, the sum of ONL1-INPRES was 18 and 19 for ONL2-INPRES.

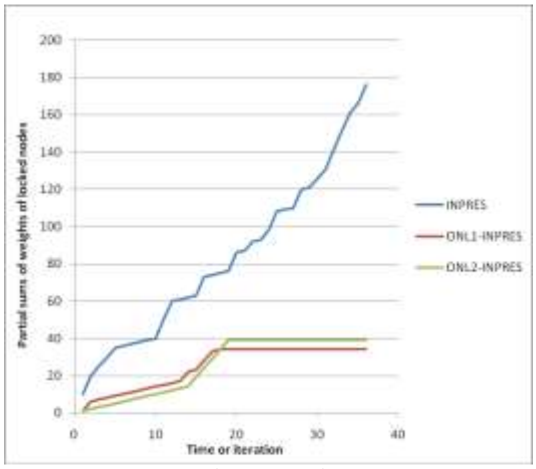

Figure 11 - System modifications performed by each algorithm

In the previous graph we can observe the behaviour we described in the first experiment, where ONL2 produced a total sum of weights of locked nodes higher than ONL1 despite it began with a partial sum lower than its counterpart.

4.3 Experiment 3 - arbitrary system

This last experiment consists of an arbitraty system, shown in Fig. 13.

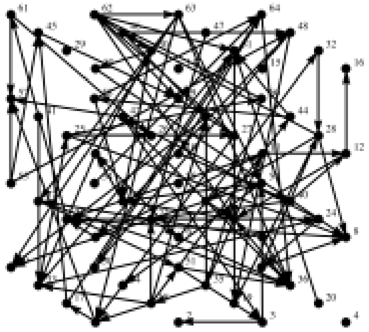

Figure 12 – Topology of experiment 3

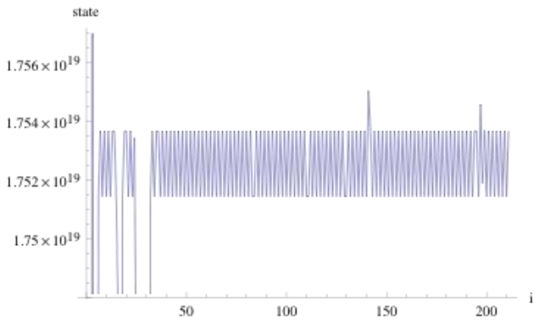

Figure 13 – In experiment 3, the system was unstable.

INPRES	{1,1,1,1,1,0,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1, 0,1,1,0,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,0}
ONL1-INPRES	{0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0}
ONL2-INPRES	{0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0}

Table 3 - Resulted locking vectors per algorithm, experiment 3

The locking vector for ONL1-INPRES and ONL2-INPRES are the same. However, the algorithms did not lock the nodes in the same order. In the following graph we can observe the stability achieved by both ONL1-INPRES and ONL2-INPRES.

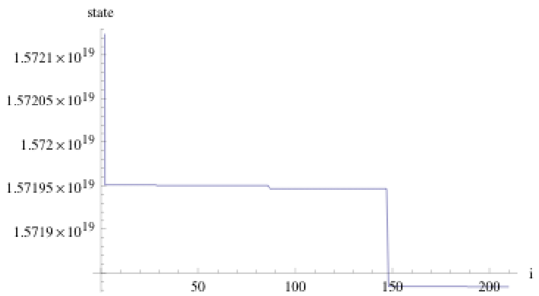

Figure 14 - Stability with ONL1-INPRES and ONL2-INPRES

INPRES locked 55 nodes, whereas ONL1-INPRES and ONL2-INPRES both locked 5 nodes. The sum of the weights of the locked nodes in INPRES was 262 and the sum of the weights for both ONL1-INPRES and ONL2-INPRES were 9.

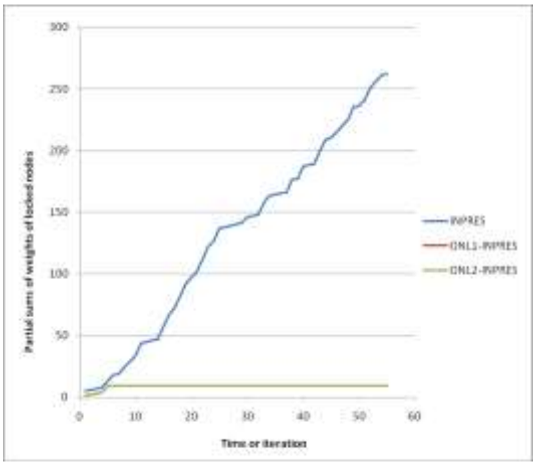

Figure 15 - System modifications performed by each algorithm

This experiment clearly shows the benefits of using the algorithms in a dense and coupled system. With only a few nodes locked, the system can be brought to a stable state whereas INPRES needed many more locked nodes to achieve stability.

We observe that ONL1 and ONL2 produced equal numbers of locked nodes and total sums of weights in locked nodes, however, if we look closer to the order in which the nodes were locked, we can realize that the results are not totally equal. ONL1 locked the system in the order {10,17,30,33,59}, whereas ONL2 locked the environment in the order {30,33,17,10,59}.

5. DISCUSSION

	INPRES		ONL1-INPRES		ONL2-INPRES	
Experiment number	# locked nodes	Σ locked nodes	# locked nodes	Σ locked nodes	# locked nodes	Σ locked nodes
1	44	253	16	44	16	44
2	36	176	18	34	19	39
3	55	262	5	9	5	9
4			5	13	8	16
5			6	10	9	13
6			7	16	9	18

Table 1 – Summary table, performance comparison between algorithms

5.1 INPRES vs. ONL-INPRES

As we can see in the previous table and the experiments presented, in general, both ONL1-INPRES and ONL2-INPRES produced better results than INPRES, both in the number of locked nodes as in the total sums of weights of locked nodes.

In experiment 1, ONL1-INPRES and ONL2-INPRES locked just 36.6% of the number of nodes that INPRES did. In experiment 3, the new algorithms locked just 9% of the nodes in comparison to INPRES. In experiment 3, ONL-INPRES' algorithms accounted for just the 3.4% of the total sum of weights in locked nodes in comparison to INPRES.

It is clear that applying these algorithms in a real-time environment would lead to a much less-disabled system while avoiding cyclic instabilities in the system.

5.2 ONL1-INPRES vs. ONL2-INPRES

5.2.1 Number of locked nodes

The greater the density and coupling of the system, the less nodes the algorithms need to achieve stability. This is actually logical: it only needs to lock a few nodes to lock all the cycles of the system (because they are so interconnected).

With the experiments presented, one can observe that both algorithms are producing a similar number of locked nodes. In some cases, ONL1 and ONL2 produced exactly an equal number of locked nodes (experiment 1 and 3) and in other experiments, ONL2 locked a slightly higher number of nodes (experiment 2, 4, 5 and 6).

The difference between ONL2 and ONL1 is that the former tries to pick the less weighted nodes overall. However, because the system is dense and very coupled, when any of the two algorithms decide to lock a node, they affect the system in a considerable way, meaning that many of the cycles will have the node that the algorithms decided to lock in the first place, so more than a few cycles will be stabilized by deciding to lock that one first node.

In other words, what ONL2 is doing is trying to find the best node in the system to start locking. Experiments have pointed out that it really does not matter where the algorithms start locking, the system is so dense and coupled that the result will be very similar as picking the first node of the system (as ONL1 does). This leads us to think that the configuration space of the system is, in this sense, isotropic. However, this does not mean that the algorithms will pick the same nodes, it just means that the number of picked or locked nodes, tend to be equal (as shown in experiment 3).

5.2.2 Total sum of weights of locked nodes

The results of the presented experiments have a tendency to point out that the algorithms are likely to produce similar total sums of weights in locked nodes. We believe this is a behavior that arises due to the conjunction of some other circumstances.

First, we must realize that the algorithms tend to produce an equal number of locked nodes to stabilize the environment (as stated before).

The set of possible weights in the experiments performed consists of 3 possible values {1, 5, 10}. At running time, ONL1 picks the first cycle it finds, and then finds the minimum weighted node in that cycle. Thus, we can realize that there is a high probability that that first node will be a node with a weight of 1 (the probability of having a 1, 5 or 10 as a weight, is equal). ONL 2 tries to pick the best nodes overall (the ones with less weights). Therefore, the best ONL2 will be able to do, is pick a node with a weight of 1 (just the same as ONL1), and because the algorithms are likely to lock the same number of nodes, we can realize why they also tend to produce a similar total of weights of locked nodes (even though ONL2 was designed to perform better in this objective). We believe that a non-homogeneous allocation of the weights for the nodes would allow ONL2-INPRES to exhibit a higher performance in comparison to ONL1, for the reasons previously mentioned.

6. Conclusions and Future Work

In this research we analyzed experimentally two algorithms, ONL1-INPRES and ONL2-INPRES. These two algorithms have been proven to find a set of nodes to lock, in order to eliminate cyclic behaviour. These algorithms not only stabilize the system, but also minimize the number of nodes locked (minimizing the loss of functionality of the system) and total weight of the nodes locked (impacting the less important agent in the system). These are clearly very important results in terms of the services provided to the user.

Additionally, the experimental results showed that the two algorithms –one focused on minimizing the number of nodes locked, and the other on minimizing the total weight of nodes locked- performed in a very similar way, as it can be seen on table 7.

ONL1 and ONL2 performed much better compared to INPRES. Also, from the previous analysis it has been found that ONL1 and ONL2 performed in a very similar way, despite the fact that ONL2 should have achieved better results, as it was designed to minimize the weight of the locked nodes. One possible explanation for this is that the order of the locking process is not important. In this sense, the configuration space is isotropic (in the number of nodes): for medium and high density systems, it is not important which nodes are locked first, as in the long run the two algorithms will lock the same number of nodes. However, more research is needed in this direction.

Paradoxically, for very high densities, the tendency is to lock fewer nodes, due to the high coupling of the cycles. In the extreme case of a fully connected system, only one node should be locked. However, on the other hand, probably, the system wouldn't oscillate at all, due to the multiple restrictions imposed by the coupled rules. This behaviour of the locking strategy could be used in order to estimate the degree of coupling for a given system. More research is needed in this direction.

For future work, we will continue to experiment with these algorithms and more specifically with non-homogeneous allocations of weights in the system.

Finally, our experiments have shown an efficient way of stabilizing the system. This involves finding the nodes which are part of the most cycles and which are less weighted overall. Based on what we have learnt, we expect this would lead to the most-efficient way of stabilizing the system.

The results presented in this paper are of great importance. The efficiency achieved and the impact they would have in a real multi agent system are considerable, much better than previous work. Furthermore, we believe questions and directions pointed out in this paper are of great value for future research in multi-agent based ambient intelligence and related fields.

7. ACKNOWLEDGMENT

The authors want to thank ITESM and ITL for the support provided to this work.

8. REFERENCES

[1] Are Cars Too Complicated?, http://spectrum.ieee.org/podcast/green-tech/advanced-cars/are-cars-too-complicated

[2] Callaghan V, Colley M, Hagras H, Chin J, Doctor F and Clarke G, Programming iSpaces: a tale of two paradigms, in Steventon, A., Wright, S. (Eds.), Intelligent Spaces: The Application of Pervasive ICT part of the series Computer Communications and Networks, Springer, Heidelberg, Germany. pp. 162. 2005.

[3] Chin JS, Callaghan V and Clarke G, An End-User Programming Paradigm for Pervasive Computing Applications, The IEEE International Conference on Pervasive Services, Lyon, France, June 26-29, 2006 Page(s):325 – 328.

[4] De Carolis, B. and Cozzolongo, G. (2004) 'C@sa: intelligent home control and simulation', *Internat. J. Comput. Intelligence*, Vol. 1, No. 1, pp. 1-12.

[5] Hagras H, Callaghan V, Colley M, Clarke G, Pounds-Cornish A and Duman H, Creating an ambient-intelligence environment using embedded agents. IEEE on Intelligent Systems, Volume 19, Issue 6, Nov-Dec 2004 Page(s):12 – 20.

[6] Henricksen K, Indulska J, Rakotonirainy A, Modeling context information in pervasive computing systems, Proceedings of the First International Conference on Pervasive Computing, Zurich, Switzerland, August 26-28 2002, pg 167. Springer (2002).

[7] Holloway, Eric M., Lamont, Gary B., Self organized multi-agent entangled hierarchies for network security, Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. Montreal, Quebec, Canada. Workshop Session: Evoutionary computation and multi-agent systems and simulations. Page: 2589-2596, 2009.

[8] Milner R., The Space and Motion of Communicating Agents. Cambridge University Press, 1st edition, 2009.

[9] Razzaque MA, Dobson S and Nixon P, Categorisation and Modelling of quality in context information, in Proceedings of the IJCAI 2005 Workshop on AI and Autonomic communications. Roy Sterrit, Simon Dobson and Mikhail Smirnov (ed). 2005.

[10] Rohlf, T, Jost, J., A new class of cellular automata: How spatio-temporal delays affect dynamics and improve computation. European Conference on Complex Systems. University of Warwick, United Kingdom. September 21-25 2009.

[11] Shehata M., Eberlein A., Fapojuwo A., Mohamed A. (2007) Managing Policy Interactions in KNX-based Smart Homes, 1st IEEE International Workshop on Development and Application of Knowledge-Based Software Tool (KASET), Proceedings of the 31st Annual IEEE International Computer Software and Applications Conference (COMPSAC 2007), July 23-27, 2007, Beijing, China

[12] Shehata, M., Eberlein, A., and Fapojuwo, A. 2007. Using semi-formal methods for detecting interactions among smart homes policies. *Sci. Comput. Program.* 67, 2-3 (Jul. 2007), 125-161. DOI= http://dx.doi.org/10.1016/j.scico.2006.11.002

[13] Strogatz SH, Exploring Complex Networks, Nature 410, 268-276. 2002.

[14] Weisbuch G, *Complex Systems*. Lecture Notes Volume II. Santa Fe Institute Studies in the Sciences of Complexity. 1991.

[15] Zamudio V and Callaghan V. Facilitating the Ambient Intelligence Vision: a Theorem, Representation and Solution for Instability in Rule-Based Multi-Agent Systems. Special Section on Agent Based System Challenges for Ubiquitous and Pervasive Computing. International Transactions on Systems Science and Applications. Vol. 4, No. 2, May 2008. pp. 108-121. Guest Editor: . J. Gaber

[16] Zamudio V and Callaghan V. Understanding and Avoiding Interaction Based Instability in Pervasive Computing Environments. International Journal of Pervasive Computing and Communications, Vol. 5 Issue 2, 2009. Page: 163-186. Guest Editors: Evi Syukur and Javier Garcia-Villalba.

[17] Zamudio V, PhD Dissertation. University of Essex. 2009.

[18] Victor Zamudio, Rosario Baltazar, Miguel Casillas, Vic Callaghan. *c-INPRES: Coupling Analysis Towards Locking optimization in Ambient Intelligence. The 6th International Conference on Intelligent Environments IE10. 19-21 July 2010, Monash University (Sunway campus), Kuala Lumpur, Malaysia. Page(s) 68-73. Editors Vic Callaghan, Achilles Kameas, Simon Egerton, Ichiro Satoh, Michael Weber.*

[19] Gershenson, C. Classification of Random Boolean Networks In Standish, R. K., M. A. Bedau, and H. A. Abbass (eds.) *Artificial Life VIII: Proceedings of the Eight International Conference on Artificial Life.* . pp. 1-8. Sydney, Australia. MIT Press. 2002.